

# PATENT ABSTRACTS OF JAPAN

(11)Publication number : 09-152961

(43)Date of publication of application : 10.06.1997

(51)Int.Cl. G06F 9/06  
G06F 9/445

(21)Application number : 08-195380

(71)Applicant : SUN MICROSYST INC

(22)Date of filing : 05.07.1996

(72)Inventor : EVANS RODRICK I  
GINGELL ROBERT A

(30)Priority

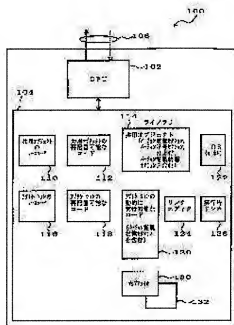
Priority number : 95 499062 Priority date : 06.07.1995 Priority country : US

## (54) METHOD FOR IMPARTING VERSION SETTING INFORMATION TO SOFTWARE PROGRAM AND DEVICE THEREFOR

(57)Abstract:

**PROBLEM TO BE SOLVED:** To enable an easy and efficient version setting by confirming whether the object for which a version setting is performed is possible to be utilized as an execution time linker.

**SOLUTION:** At the time of an execution, an execution time linker 126 executes an object 120 which is possible to be dynamically executed by preparing a processing file 122 and executing the file when all the versions of a shared object 114 that the object 120 requires. Thus, a link editor 124 decides which version of the shared object is required for an application program. The execution time linker 26 maps the objects of the memory and couples the objects. Thus, the execution time linker 126 couples the objects just as the linker 26 is instructed by the link editor 124. Further, when a right version does not exist, the execution time linker 126 generates an error.



## LEGAL STATUS

[Date of request for examination]

[Date of sending the examiner's decision of rejection]

[Kind of final disposal of application other than the examiner's decision of rejection or application converted registration]

[Date of final disposal for application]

[Patent number]

[Date of registration]

[Number of appeal against examiner's decision of rejection]

[Date of requesting appeal against examiner's decision of rejection]

[Date of extinction of right]



【特許請求の範囲】

【請求項1】 ソフトウェアプログラムにバージョン設定情報を付す方法であって、ソフトウェアプログラムのためのオブジェクトコードを用意するステップと、

前記ソフトウェアプログラムのバージョンに関するバージョン名を示すマップファイルを用意するステップと、前記ソフトウェアプログラムのバージョン名を示す情報が、前記マップファイルに従って前記オブジェクトコードに付加されるよう、前記オブジェクトコードをリンクし、これにより、バージョン設定されたオブジェクトを生成するステップとを具備し、これらのステップがデータ処理システムによって実行される方法。

【請求項2】 ソフトウェアプログラムにバージョン設定情報を付す方法であって、

第1のソフトウェアプログラムのための第1のオブジェクトコードを用意するステップと、前記第1のソフトウェアプログラムのバージョンに関するバージョン名を示すマップファイルを用意するステップと、

前記第1のソフトウェアプログラムのバージョン名を示す情報が、前記マップファイルに従って前記第1のオブジェクトコードに付加されるよう、前記第1のオブジェクトコードをリンクし、これにより、バージョン設定されたオブジェクトを生成するステップと、

第2のソフトウェアプログラムのための第2のオブジェクトコードを用意するステップと、

前記第2のオブジェクトコードを前記バージョン設定されたオブジェクトにリンクするステップであって、このステップが、前記第2のソフトウェアプログラムに必要なとされる前記第1のソフトウェアプログラムのバージョンを判定するステップと、前記第2のソフトウェアプログラムに必要なとされる前記バージョンを示す情報を前記第2のオブジェクトコードに付加し、これにより、動的に実行可能なプログラムを生成するステップとをさらに含むものと、を具備し、これらのステップがデータ処理システムによって実行される方法。

【請求項3】 ソフトウェアプログラムにバージョン設定情報を付す装置であって、

第1のソフトウェアプログラムのための第1のオブジェクトコードを格納する記憶媒体と、

前記第1のソフトウェアプログラムのバージョンに関するバージョン名を指定するマップファイルを格納する記憶媒体と、

前記マップファイルに従って、前記第1のオブジェクトコードに対して、前記第1のソフトウェアプログラムの前記バージョンのバージョン名を定義する付加情報を付し、これにより、バージョン設定されたオブジェクトを生成するリンクと、

第2のソフトウェアプログラムのための第2のオブジェ

クトコードを格納する記憶媒体と、

前記第2のソフトウェアプログラムに必要なとされるバージョンを示す付加情報を前記第2のオブジェクトコードに付すことによって、前記第2のオブジェクトコードを前記バージョン設定されたオブジェクトにリンクし、これにより、動的に実行可能なプログラムを生成するリンクとを具備した装置。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】 この発明は、ソフトウェアプログラムをリンクする方法および装置に関し、特に、ソフトウェアプログラムの連続したバージョンの変更を管理する動的リンクシステムを提供する方法および装置に関する。

【0002】

【従来の技術】 ソフトウェア開発というものは絶えず現在進行中のプロセスといえるものである。ソフトウェアの最初のバージョンは、該ソフトウェアが書かれたときのタスクには十分であるが、時間が経過し、新たな特徴が追加されるに伴って、アップグレードを必要とする。ソフトウェアアプリケーションが共用オブジェクト（“ライブラリ”とも言う）に結合（バインド）されるような場合、このソフトウェア開発プロセスは、特に問題を伴う。共用オブジェクトが更新または変更される場合、しばしば、該共用オブジェクトに対するインターフェイスも更新または変更される。さらに、共用オブジェクトに対するインターフェイスが変更されない場合でも、しばしば、前記共用オブジェクトによって実行される機能の一部に変化が生じる。

【0003】

【発明が解決しようとする課題】 従来のシステムのあるものは、実行時（ランタイム）に、アプリケーションソフトウェアプログラムを共用オブジェクトに動的にリンクする。このようなシステムにおいては、共用オブジェクトの新たなバージョンがリリースされる度に、該共用オブジェクトにアクセスするアプリケーションソフトウェアプログラムを慎重にチェックする必要がある。また、（前記共用オブジェクトに対するインターフェイスが同じであっても）前記共用オブジェクトの動作が変わっていないか否かを判定するためにも、前記アプリケーションソフトウェアプログラムをチェックする必要がある。従来、このようなチェックは人手によって行われていた。インターフェイスの不整合によって生じるエラーは、しばしば、インターフェイスの不整合が見つかり、または、共用オブジェクトがそれまでと同じ動作を行わないことにより、アプリケーションソフトウェアプログラムの実行中にのみ発見される。このような場合必要なのは、特定のアプリケーションソフトウェアプログラムが共用オブジェクトのどのバージョンにリンクしようとするのかを判定する手段、および、前記共用オブジェ

トに必要なバージョンが実行時における動的リンク処理中に存在しているか否かをチェックする手段である。

【0004】従来、前記共用オブジェクト自体のファイル名は、新たなバージョンごとに更新されていた。このため、リンク処理中には前記共用オブジェクトの最も新しいバージョンのみが存在し、この最も新しいバージョンは該オブジェクトの古いバージョンとは完全に異なるファイル名を有することになる。この点につき、オブジェクトの新たなバージョンが作成される毎に該オブジェクトの名前を変えないようにすることが望ましい。従来のシステムは、しばしば、アプリケーションプログラムおよび共用オブジェクト（ライブラリ）を“1かたまりのシステム”としてリリースすることによって、バージョンチェックを回避しようとしている。すなわち、共用オブジェクトのバージョンの変更がなされたか否かに関わらず、新たなアプリケーションは、それが必要とするオブジェクトのすべてと共に出荷される。この点につき、前記システムにおける、バージョン変更に対処するために必要な部分のみをアップグレードできるようにすることが望ましい。

【0005】この発明は上述の点に鑑みてなされたもので、オブジェクトの新たなバージョンが作成される毎に該オブジェクトの名前を変えないようにすることができ、且つ、バージョン変更に対処するために必要な部分のみをアップグレードできるようにすることにより、簡単かつ効率的なバージョン設定を可能にする、ソフトウェアプログラムにバージョン設定情報を付す方法および装置を提供することを目的とする。

【0006】

【課題を解決するための手段】この発明は、ソフトウェアプログラムを動的にリンクする方法および装置を提供するものであり、また、アプリケーションソフトウェアプログラムによってアクセスされるオブジェクトの連続したバージョンにおける変更を把握するバージョンシステムを提供する。この発明は、バージョン設定されたオブジェクトにアクセスするためにアプリケーションソフトウェアプログラムによって使用されるインターフェースをチェックし、前記アプリケーションソフトウェアプログラムによる前記バージョン設定されたオブジェクトの無効バージョンにアクセスする試みを検出する。こうして、この発明は、アプリケーションソフトウェアプログラムとオブジェクトとの間の後方互換性を維持しながら、オブジェクトの制御された進化を可能にするものである。

【0007】この発明は、記号インターフェースおよび実行上の変更が1つのオブジェクト内においてラベル表示できるようにする、バージョン設定システムを提供する。構築時において、リンクエディタは、バージョン設定されたオブジェクトに対して、該オブジェクトのすべての利用可能なバージョンを定義するデータ（バージョ

ン定義セクションおよびバージョン記号セクション）を付加する。また、構築時において、リンクエディタは、ソフトウェアアプリケーションに対して、該アプリケーションのバージョン要件を定義するデータ（バージョン従属状態セクション）を付加する。実行時において、実行時リンカは、前記ソフトウェアアプリケーションの要件が前記オブジェクト自体に格納されたバージョン定義に一致するか否か、すなわち、前記ソフトウェアアプリケーションに必要とされる前記バージョン設定されたオブジェクトが前記実行時リンカに利用可能であるか否かを確認する。

【0008】バージョンとは、オブジェクトに記録された名前またはラベルである。バージョンは1つまたは2つ以上の大域記号（グローバルシンボル）に関連づけられてよく、この場合、バージョンは記号インターフェースを定義する。そうでない場合、バージョンは、単に、実行上の変更、すなわち、オブジェクトの機能の変更が存在するものの、新たな大域記号の定義はなされない変更を示す標識であってよい。後者の場合、そのバージョンは“弱い（ウィーク）”バージョンと呼ばれる。ここに説明するこの発明の実施の形態において、各バージョンに関連した大域記号およびバージョン名は、人によって発生される“マッピングファイル”に定義される。バージョン設定されたオブジェクトを作成するために、前記マッピングファイルは、構築時に、1つまたは2つ以上の再配置可能な（コンパイルされた）オブジェクトと共に前記リンクエディタに入力される。構築時に、アプリケーションが（バージョン設定情報を有する）バージョン設定されたオブジェクトとリンク編集されるとき、デフォルト値により、前記アプリケーションによって参照される大域記号を含むバージョンに対する従属状態が前記アプリケーションに設定される。さらに、弱い定義に対する“弱い”従属状態が設定される。

【0009】この発明は、1つのオブジェクト内におけるバージョン定義の継承を可能にする。バージョンはバージョンを受け継ぎ、これにより、相互に関連したインターフェース定義を作成するために、複数組の記号を組み合わせることができる。例えば、1つの新たなバージョンは、古いバージョンのすべての大域記号を受け継ぐことができる。この発明は、リンクエディット時におけるオブジェクトのバージョンの可視性を制御でき、実際上、前記アプリケーションプログラムに利用可能なインターフェースを制御できる。また、この発明は、前記アプリケーションプログラムがオブジェクトの弱いバージョンを要求することを強制することもできる。

【0010】上記目的を達成するため、この発明は、ソフトウェアプログラムにバージョン設定情報を付す方法であって、ソフトウェアプログラムのためのオブジェクトコードを用意するステップと、前記ソフトウェアプログラムのバージョンに関するバージョン名を示すマッ

ファイルを用意するステップと、前記ソフトウェアプログラムのバージョン名を示す情報が、前記マップファイルに従って前記オブジェクトコードに付加されるよう、前記オブジェクトコードをリンクし、これにより、バージョン設定されたオブジェクトを生成するステップとを具備し、これらのステップがデータ処理システムによって実行されるものである。これにより、この発明によれば、バージョン設定されたオブジェクトを自動的に生成することができる。

【0011】さらに、この発明は、ソフトウェアプログラムにバージョン設定情報を付す方法であって、第1のソフトウェアプログラムのための第1のオブジェクトコードを用意するステップと、前記第1のソフトウェアプログラムのバージョンに関するバージョン名を示すマップファイルを用意するステップと、前記第1のソフトウェアプログラムのバージョン名を示す情報が、前記マップファイルに従って前記第1のオブジェクトコードに付加されるよう、前記第1のオブジェクトコードをリンクし、これにより、バージョン設定されたオブジェクトを生成するステップと、第2のソフトウェアプログラムのための第2のオブジェクトコードを用意するステップと、前記第2のオブジェクトコードを前記バージョン設定されたオブジェクトにリンクするステップであって、このステップが、前記第2のソフトウェアプログラムに必要とされる前記第1のソフトウェアプログラムのバージョンを判定するステップと、前記第2のソフトウェアプログラムに必要とされる前記バージョンを示す情報を前記第2のオブジェクトコードに付加し、これにより、動的に実行可能なプログラムを生成するステップとをさらに含むもの、とを具備し、これらのステップがデータ処理システムによって実行されるものである。これにより、この発明によれば、バージョン設定されたオブジェクトを自動的に生成することができると共に、第2のソフトウェアプログラムすなわちアプリケーションソフトウェアプログラムを、このバージョン設定されたオブジェクトに動的にリンクして実行可能とすることができる。

【0012】別の観点に従えば、この発明は、ソフトウェアプログラムにバージョン設定情報を付す装置であって、第1のソフトウェアプログラムのための第1のオブジェクトコードを格納する記憶媒体と、前記第1のソフトウェアプログラムのバージョンに関するバージョン名を指定するマップファイルを格納する記憶媒体と、前記マップファイルに従って、前記第1のオブジェクトコードに対して、前記第1のソフトウェアプログラムの前記バージョンのバージョン名を定義する付加情報を付し、これにより、バージョン設定されたオブジェクトを生成するリンクと、第2のソフトウェアプログラムのための第2のオブジェクトコードを格納する記憶媒体と、前記第2のソフトウェアプログラムに必要とされるバージョ

ンを示す付加情報を前記第2のオブジェクトコードに付すことによって、前記第2のオブジェクトコードを前記バージョン設定されたオブジェクトにリンクし、これにより、動的に実行可能なプログラムを生成するリンクとを具備したものである。

【0013】別の観点に従えば、この発明は、動的に実行可能なオブジェクトに必要とされるオブジェクトのバージョンが該実行可能オブジェクトの実行中に存在することを判定させるための、コンピュータによって読み取り可能なコードを格納したコンピュータによって使用可能な媒体を備えたコンピュータプログラム製品であって、コンピュータに、第1のソフトウェアプログラムのための第1のオブジェクトコードを用意させる第1のコンピュータ読み取り可能プログラムコード装置と、コンピュータに、前記第1のソフトウェアプログラムのバージョンに関連したバージョン名を指定するマップファイルを用意させる第2のコンピュータ読み取り可能プログラムコード装置と、前記第1のソフトウェアプログラムのバージョン名を示す情報が、前記マップファイルに従って、前記第1のオブジェクトコードに付加されるよう、コンピュータに、前記第1のオブジェクトコードをリンクさせ、これにより、バージョン設定されたオブジェクトを生成させる第3のコンピュータ読み取り可能プログラムコード装置とを具備したものである。

【0014】

【発明の実施の形態】以下、添付図面を参照してこの発明の一実施の形態を詳細に説明する。

#### 1. 概論

図1は、この発明に係るコンピュータシステム100のブロック図である。該コンピュータシステム100は、CPU102と、メモリ104と、入出力ライン106とを備えている。当業者に理解されるように、前記コンピュータシステム100は、ディスクドライブ、キーボード、ディスプレイ装置、ネットワーク接続部、付加的なメモリ、付加的なCPU等、ここでは明確のために図示していないその他の多数の要素を含んでいてもよい。前記メモリ104は、ライブラリ（共用オブジェクトとも言う）114と、該共用オブジェクト114のソースコード110および再配置可能な（コンパイルされた）コード112とを格納している。さらに、前記メモリ104は、アプリケーションソフトウェアプログラムのソースコード116と、アプリケーション116の再配置可能な（コンパイルされた）コード118と、アプリケーション116の動的に実行可能な（リンクされた）コード120とを格納している。また、前記メモリ104は、OS（カーネル）ソフトウェア122と、マップファイル130、132と、リンクエディタ124と、実行時（ランタイム）リンク126とを格納している。前記共用オブジェクト114および実行可能オブジェクト120の各々は、前記リンクエディタ124によ

って作成されるものである。前記共用オブジェクト114および実行可能コード120の各々は、Prentice Hall, Inc.によって発行された“System V Application Binary Interface”の第3版に定義されたExecutable Linking Format (実行可能リンクフォーマット、以下、“ELFフォーマット”とも言う)と同様なELFを有する。

【0015】しかしながら、以下に説明するように、この発明において、前記オブジェクト114、120のELFフォーマットは、付加的なデータを含むよう拡張されている。共用オブジェクト114は、図9に示すフォーマットを有する。前記動的に実行可能なオブジェクト120は、図14に示すフォーマットを有する。以下に説明するように、前記共用オブジェクト114は、該共用オブジェクト114の各バージョン毎のバージョン設定情報(バージョン定義セクション)と、各バージョン毎の公用記号リスト(バージョン記号セクション)とを含んでいる。さらに、前記共用オブジェクト114は、バージョンの従属状態(関係)に関する情報(バージョン従属状態セクション)を含んでいてよい。実行可能オブジェクト120は、バージョンの従属状態(関係)に関する情報(バージョン従属状態セクション)を含んでいる。明確さのために図示されていないが、当業者に理解されるように、前記メモリ104は、アプリケーションプログラム、オペレーティングシステム、データ等のその他の情報をも格納している。

【0016】この発明の好ましい実施の形態は、Solarisオペレーティングシステムのバージョン2.5の下に実施される。SolarisはSun Microsystems, Inc.の登録商標である。また、Unixは、X/OPEN, Ltd.により排他的にライセンスされ、米国およびその他の国で登録された商標である。図2は、図1のリンクエディタ124の入出力を示す図であり、バージョン設定された共用オブジェクトの作成を示す。以下では共用オブジェクトのバージョン設定について説明するが、この発明は、動的に実行可能なオブジェクトおよび再配置可能なオブジェクトのバージョン設定を実行するためにも使用可能である。このように、これらの種類のオブジェクトは、バージョン定義セクションおよびバージョン記号セクションを含むことができる。前記リンクエディタ124は、マップファイル130および共用オブジェクト用の再配置可能なオブジェクトコード112の入力を受け取り、共用オブジェクト114の入出力を発生する。前記マップファイル130は、前記共用オブジェクトの各バージョン毎に大域(グローバル)記号およびバージョン名を指定する。ここに説明する実施の形態において、好ましくは、前記マップファイル130は、図8のフォーマットを有し、人によって作成される。他の実施の形態において、前記マップファイル130は、コンパイルシステムによって作成されてもよい。

【0017】図3は、図1のリンクエディタ124の他の入出力を示す図であり、図2の共用オブジェクトによるアプリケーションプログラムのリンクエディタ(編集)を示すものである。前記リンクエディタ124は、共用オブジェクト114および再配置可能なオブジェクト118を入力し、これらのオブジェクト114、118を処理することによって動的に実行可能なオブジェクト120を生成する。図4に示すように、前記オブジェクト114のどのバージョンがこのリンク手続に許容されるかを示すために、このステップにマップファイル132が使用されてもよい。好ましくは、該マップファイル132は、図3のフォーマットを有し、人によって作成される。

【0018】図5に示すように、実行時において、実行時リンカ126は、動的に実行可能なオブジェクト120が必要とする共用オブジェクト114のすべてのバージョンが存在するか否かを確認する。前記すべてのバージョンが存在する場合、前記実行時リンカ126は、処理ファイル122を作成して実行することによって、動的に実行可能なオブジェクト120を実行する。こうして、前記リンクエディタ124は、再配置可能なオブジェクトからバージョン設定された共用オブジェクトを構築し、前記共用オブジェクトのどのバージョンがアプリケーションプログラムに必要とされるのかを判定する。前記実行時リンカ126は、そのメモリのオブジェクトをマップし、結合する。こうして、前記実行時リンカ126は、単に、前記リンクエディタ124によって指示されるようにオブジェクトを結合する。さらに、前記実行時リンカ126は、動的に実行可能なオブジェクト120が必要とする共用オブジェクト114のバージョンが存在することを保証するための、確認チェックを行う。正しいバージョンが存在しない場合、前記実行時リンカ126はエラーを発生する。

【0019】次に、共用オブジェクト114の様々なバージョン間で可能な変更について説明する。一般に、これらの変更は、互換性のある更新と互換性の無い更新との2つのグループに分類できる。前記互換性のある更新とは、付加的な更新、すなわち、共用オブジェクト114に対するインターフェースでそれまで利用可能であった大域記号がそのまま維持される更新である。互換性のある更新の一例は、大域記号の付加である。前のバージョンから記号は除去されていないので、前のバージョンとインターフェース接続されたアプリケーションソフトウェアは依然として正しく動作する。前記互換性の無い更新は、特定の既存のインターフェースを使用した既存のアプリケーションが故障したり不正動作したりするものである。このような互換性の無い更新の例としては、記号の除去、機能に対する引数の付加、機能からの引数の除去、および、機能に対する引数のサイズまたは内容

の変更などがある。共用オブジェクト 114 に対する限り訂正は、既存のインターフェースに対する互換性のある更新であったり、互換性の無い更新であったりする。例えば、互換性のある限り訂正は、単に、それまで定義されていたインターフェースを維持しながら、共用オブジェクト 114 の内部機能を変更するものである。一方、互換性の無い限り訂正は、共用オブジェクト 114 に対するインターフェースの変更を必要とする。

【0020】2. 構築時におけるバージョン設定情報の作成

上記パラグラフは、この発明に従うバージョン設定を実行するために、構築時および実行時に行われる処理の全体的な説明であった。以下のパラグラフは、前記リンクエディタ 124 が、どのようにして、共用オブジェクトおよびアプリケーションプログラムにバージョン設定情報を付加するのかを説明するものである。

【0021】a. 構築時にバージョン設定されたオブジェクトに関する

バージョン定義作成

以下に説明するように、好ましい実施の形態において、前記リンクエディタ 124 は、マップファイル 130 のバージョン指示情報に従って、共用オブジェクトのバージョン設定を制御する。前記マップファイル 130 は、人によって作成されるのが好ましいが、コンパイルシステムのようなソフトウェアによって作成されてもよい。図 6 および図 7 は、図 2 のリンクエディタ 124 によって実行されるステップを示すフローチャートである。これらのステップは、共用オブジェクト 114 のようなバージョン設定されたオブジェクトを作成するために実行されるステップの一部である。当業者によって理解されるように、図 6 および図 7 (および図 12) のステップは、メモリ 104 に格納されていて、且つ、例えばメモリ 104 に格納されたデータ構造を使用したリンクエディタ 124 の命令を実行する図 1 の CPU 102 によって行われる。

【0022】ステップ 302 に示すように、図 6 のステップは、リンクエディタ 124 がコマンドライン上の G オプションによって起動された時に開始される。前記 G オプションは、リンクエディタ 124 が (動的に実行可能なオブジェクトとは対照的な) 共用オブジェクトを生成すべきことを示すものである。この発明の好ましい実施の形態において、前記リンクエディタ 124 は、M オプションによっても起動される。該 M オプションは、マップファイル 130 が “バージョン定義指示” ソースとして使用されるべきことを示すものである。図 2 の例は共用オブジェクトの作成を示しているが、この発明は、バージョン設定された再配置可能なオブジェクトおよび動的に実行可能なオブジェクトを作成するためにも使用可能である。

【0023】以下の例は、使用される Unix コマンド (ca

t, cc, pvs, ld) の詳細を含んでいない。該 Unix コマンドは、Sun Microsystems から入手可能な Solaris Reference Manual に記載されている。この後のパラグラフでは、共用オブジェクトソースコード 110 およびマップファイル 130 の一例について説明する。表 1 は、C プログラミング言語で書かれた 4 つのソースファイル (“foo.c”, “data.c”, “bar1.c” および “bar2.c”) のソースコード 110 を示している。これらのソースコードファイルは、1 つの共用オブジェクトを構成している。また、表 2 のマップファイルは、前記共用オブジェクトの様々なバージョンの大域インターフェースを定義している。これらのファイルは、コンパイルされることにより、図 1 の再配置可能なオブジェクト 112 を構成する。その後、前記リンクエディタ 124 は、後で説明するように、マップファイル 130 に従って、バージョン設定された共用オブジェクト 114 を作成する。

【0024】

【表 1】

```
$ cat foo.c
extern const char * foo1;
extern const char * _foo2;
void foo1()
{
    (void) printf("foo1");
}
void foo2()
{
    (void) printf(_foo2);
}

$ cat data.c
const char * _foo1 = "string used by function foo1()\n";
const char * _foo2 = "string used by function foo2()\n";

$ cat bar1.c
extern void foo1();
void bar1()
{
    foo1();
}

$ cat bar2.c
extern void foo2();
void bar2()
{
    foo2();
}
```

【0025】

【表 2】



```

$ cat mapfile
SUNW.1.1 {                                #Release X
    global:
        fool;
    local:
        *;
};
SUNW.1.2 {                                #Release X+1
    global:
        foo2;
} SUNW.1.1;
SUNW.1.2.1 { } SUNW.1.2;                 #Release X+2
SUNW.1.3a {                               #Release X+3
    global:
        bar1;
} SUNW.1.2;
SUNW.1.3b {                               #Release X+3
    global:
        bar2;
} SUNW.1.2;
SUNW.1.4 {
    global:
        bar3;
} SUNW.1.3a SUNW.1.3b
【0026】図8は、Backus-Naur (バックスナウル)
フォーマットを使用したマップファイル130のフォー
マットを示すものである。該バックスナウルフォーマ
ットにおいて、角括弧 [ ] および " " はオプション
要素を示す。例えば、図8において、" [version nam
e] " は前記マップファイルフォーマットのオプション
要素である。共用オブジェクト114は、実行時にお
いて動的に実行可能なオブジェクト120のような他の
オブジェクトが結合することが可能な大域記号を提供す
る。これらの大域記号は、前記マップファイル130に
おいて指定されており、共用オブジェクト114のア
プリケーション・バイナリ・インターフェース (ABI)
を記述する。共用オブジェクト114の生存期間中、オ
ブジェクトのインターフェースは、大域記号の付加また
は除去によって変更可能である。さらに、共用オブジェ
クト114の進化は、前記インターフェースの大域記号
に影響しない、該オブジェクト114に対する内部実行
上の変更を伴うことがある。
【0027】表2は、マップファイル130の一例を示
すものである。該表2において、前記マップファイル1
30は、共用オブジェクト114のバージョンであるSU

```

NW.1.1, SUNW.1.2, SUNW.1.2.1, SUNW.1.3a, SUNW.1.3b  
 およびSUNW.1.4に関するバージョン定義を含んでいる。  
 これらのバージョン定義は、該共用オブジェクト114  
 についてそれまで定義されていたすべてのバージョン  
 (およびそれらの大域記号) を含む。この例において、  
 SUNW.1.1はRelease Xに含まれる共用オブジェクト114  
 のバージョンであり、SUNW.1.2は、SUNW.1.1の大域記  
 号を受け継いだ、Release X+1に含まれる共用オブジェ  
 クト114のバージョンであり、SUNW.1.2.1は、SUNW.  
 1.2の大域記号を受け継いだ、Release X+2に含まれる共  
 用オブジェクト114のバージョンである。なお、SUN  
 W.1.2.1は、新たな大域記号を含んでいないので、“弱  
 い (weak : ウィーク)”バージョンである。このパー  
 ジョンは、前記共用オブジェクトにおける“実行上”の  
 変更を示す。その他の前記バージョンは、“インターフ  
 ース”の変更を示す。SUNW.1.2.1は、前記共用オブジェ  
 クトの機能変更を有するが、インターフェース変更を有  
 さないバージョンを示す。  
 【0028】さらに、SUNW.1.3aは、SUNW.1.2の大域記  
 号を受け継いだ、Release X+3に含まれる共用オブジェ  
 クト114のバージョンであり、SUNW.1.3bも、SUNW.1.  
 2の大域記号を受け継いだ、Release X+3に含まれる共  
 用オブジェクト114のバージョンである。しかし、SUN  
 W.1.3bバージョンは、SUNW.1.3aに定義された大域記号  
 “bar1”の代りに、大域記号“bar2”を定義している。  
 SUNW.1.4は、大域記号“bar3”を定義するものであり、  
 前記SUNW.1.3aおよびSUNW.1.3bの大域記号を引き継いで  
 いる。  
 【0029】表2のテーブルの例において、記号“foo  
 1”は、SUNW.1.1バージョンの公用インターフェースに  
 定義された唯一の大域記号である。特殊な“自動縮小  
 化”命令 (“\*\*\*”) によって、前記共用オブジェクトの  
 すべてのその他の大域記号は、該共用オブジェクトに対  
 するインターフェースの一部とならないよう、局所的な  
 有効範囲に縮小される。こうして、前記SUNW.1.1パー  
 ジョンの公用インターフェースは、前記大域記号“fool”  
 に関連した、そのバージョンの内部バージョン定義によ  
 って構成される。  
 【0030】  
 【表3】

```

$ cc -o libfoo.so.1 -M mapfile -G foo.c bar1.c bar2.c data.c
$ ln -s libfoo.so.1 libfoo.so
$ pvs -dsv libfoo.so.1
libfoo.so.1:
    _end;
    _GLOBAL_OFFSET_TABLE;
    _DYNAMIC;
    _edata;
    _PROCEDURE_LINKAGE_TABLE;
    _etext;
SUNW.1.1:
    foo1;
    SUNW.1.1;
SUNW.1.2:
    foo2;
    SUNW.1.2;
SUNW.1.2.1 [WEAK]:
    SUNW.1.2.1;
SUNW.1.3a:
    bar1;
    SUNW.1.3a;
SUNW.1.3b:
    bar2;
    SUNW.1.3b;
SUNW.1.4:
    bar3;
    SUNW.1.4;

```

【0031】この表3は、表1のソースコードファイルをコンパイルし、リンクするためのUnixコマンドを示している。オブジェクトコードファイル“foo.o”、“data.o”、“bar1.o”および“bar2.o”（図示せず）は、表2のマッピングファイル130を使用して動的にリンクされ、これにより、“libfoo.so.1”と称する共用オブジェクト114を生成する。（この例の場合、ccコンパイラは、自動的にld(1)リンクエディタ124を呼出す）。コマンドライン上の-Gオプションは、リンクエディタ124が動的に実行可能なオブジェクトではなく、共用オブジェクトを生成すべきことを示す。“ln”コマンドは、ld(1)の“-l”オプションに適した“コンパイル環境”名を作成する。“pvs”Unixコマンドは、前記リンクエディタ124によって作成される共用オブジェクトのバージョン設定情報、および、各バージョンに利用可能な大域記号をプリントアウトする。

【0032】表3に示すように、前記オブジェクトについて“基準バージョン”の定義も作成される。この基準バージョンは、前記共用オブジェクト自体の名前（例えば“libfoo.so.1”）を使用して定義され、前記リンクエディタ124によって生成され確保された記号を前記オブジェクトに関連づける。例えば、表3の例において、基準バージョンの定義は、前記リンカによって作成され確保された大域記号（例えば、\_etext、\_edata、\_end、\_DYNAMIC、PROCEDURE\_LINKAGE\_TABLEおよびGLOBAL\_OFFSET\_TABLE）を含む共用オブジェクトlibfoo.so.1について作成される。

【0033】図6に戻り、(ステップ304において)バージョン名がマッピングファイル130に現れた場合、前記リンクエディタ124は、共用オブジェクト114内に、特にバージョンに関する幾つかのセクションを作成

する。これらの特別なセクションは、図9に示されており、バージョン定義セクション506、バージョン記号セクション508およびオプションのバージョン従属状態セクション510を含む。以下のパラグラフは、これらのセクションの作成および使用について説明するものである。

【0034】図10は、バージョン設定された共用オブジェクト114のバージョン定義セクション506のフォーマットを示すものである。また、図20は、表1～表3に基づくバージョン定義セクションの一例を示すものである。該バージョン定義セクションは、セクションヘッダ602（図16参照）と、構造セクション604と、フラグ606と、バージョン定義索引610と、カウント値612と、ハッシュ値614と、補助値616と、次バージョン定義ポイント618と、該バージョン自体の名前620と、該定義されたバージョンが大域記号620、622を受け継いだバージョンの名前とを含んでいる。

【0035】各共用オブジェクト114は、多数のバージョン定義を有する1つのバージョン定義セクション506を含んでいる。各フィールド604～622を、“バージョン定義”と言う。次のパラグラフは、バージョン定義の内容を説明するものである。前記セクションヘッダ602は、前記バージョン定義セクションに含まれるバージョンの数を示す。各バージョン定義セクションは、マッピングファイルに明示的に定義されていない大域記号を定義する基準バージョンに関する情報を含んでいる。このため、基準バージョンについてのフィールド604～622（バージョン定義）が、図6のステップ306で作成される。フィールド609（“基準”フラグ）は、この基準バージョン定義において設定される。

【0036】ステップ308において、前記マップファイル130に定義された共用オブジェクト114の各バージョンごとに、バージョン定義（フィールド604～622）が、リンクエディタ124によって作成される。こうして、ステップ308において、表2のマップファイル130に関して、前記基準バージョン定義の他に6つのバージョン定義が作成される。図7のステップ310～316は、ステップ306、308で作成された各バージョン定義毎に実行される。ステップ310に示すように、前記マップファイル130が1つのバージョン（例えば、表1のSUNW.1.2.1参照）に関する大域記号を定義していない場合、そのバージョン定義に関する“ウィーク”フラグ608が、ステップ312において設定される（図20参照）。

【0037】フィールド604は、その構造自体のバージョン番号である。バージョン定義索引610は、共用オブジェクト114について定義された各バージョン毎に異なる（固有の）値を有するものであり、図11との関係で後で説明することにする。カウント値612は、当該バージョンにおけるフィールド620および622の対のインスタンスの数を示すものである。ハッシュ値614は、このバージョンの名前に対するハッシュ値であり、通常のELF（実行可能フォーマット）ハッシュ関数を使用する。また、補助値616は、このバージョンの第1のフィールド620に対する索引である。次バージョン定義618は、前記バージョン定義セクションにおける次のバージョン定義のフィールド604に対する索引である。

【0038】ステップ313は、このバージョンの名前に対するエントリ620、622を作成し、このエントリを指し示すよう前記補助値616を設定する。こうして、前記第1のフィールド620は、前記バージョン自体のバージョン名を含んでいる。継承（受け継ぎ）情報は、定義されたバージョンが他のバージョンを受け継ぐ元となる1つまたは2つ以上のバージョンの名前で構成されている。図7のステップ314において、マップファイル130が図8および表2に示すように継承情報を有するか否かを調べる。ステップ316において、前記リンクエディタ124は、継承情報を保持する1つまたは2つ以上のエントリ620、622を作成する。該フィールド620、622は、特定のバージョンが他のバージョンを受け継ぐ元となるバージョン毎に存在する。前記フィールド620は受け継がれたバージョン定義の名前を含み、前記フィールド622は次のフィールド620（またはゼロ）を指し示す。

【0039】図6のステップ318において、前記リンクエディタ124は、リンク編集集中のオブジェクトのすべての記号について、バージョン記号セクションを作成する。図11は、バージョン記号セクション508のフォーマットを示す図。前記バージョン記号セクション5

08におけるエントリは、そのバージョンに関する記号テーブル内の記号と1対1に対応する（図21参照）。前記記号テーブルは、当業者によく知られており、前記System V Application Binary Interface Manualに記載されており、従って、ここでは説明しない。前記セクションヘッダ702のフォーマットは、図17との関係において説明する。前記バージョン記号セクション508のエントリ704は、記号が定義されたバージョンの索引である。表2において、SUNW.1.3bバージョンが“6”の索引610を有する場合（図20参照）、大域記号“bar2”に対応するバージョン記号セクションのエントリが“6”のエントリ値を有することになる（図21）。図11に示すように、局所的な有効範囲を有する記号のエントリは、“0”の値を有する。基準バージョン定義における記号のエントリは、“1”の値を有する。上述の如く、大域記号として明示的に定義された記号（例えば、基準セクションに関する記号、各バージョンの名前、“foo1”、“foo2”、“bar1”、“bar2”および“bar3”）のみが、バージョン記号セクション508においてゼロではないエントリ704を有する。図6のステップ320は、（必要な場合）共用オブジェクト114に関するバージョン従属状態セクションを作成する。例えば、共用オブジェクト114は、他のバージョン設定されたオブジェクトを参照することがある。このバージョン従属状態セクションの作成については、後で説明する。

【0040】b. 構築時におけるバージョン従属状態情報の作成

表4は、アプリケーションプログラムソフトウェア116（“prog.c”）のソースバージョンの一例を示すものである。“prog.c”は、共用オブジェクト1141libfoo.so.1の2つの大域記号、すなわち、“foo1”および“foo2”を参照する。これらの記号は、それぞれ、インターフェイスSUNW.1.1およびSUNW.1.2の一部として定義される。表4によると、構築時において、コンパイラccはldリンクエディタ124を起動する。構築時において、“prog.c”とバージョンSUNW.1.1、SUNW.1.2、SUNW.1.2.1間で、大域記号“foo1”および“foo2”を含む結合（バインディング）が行われる。前者の2つのバージョンは、記号結合を示す。後者のバージョンは、その弱い性質のため記録される。コンパイル/リンクコマンドによってバージョン制御指示は与えられないので、前記リンクエディタ124は、prog.cにおける大域記号を分解するときに存在する共用オブジェクト114のすべてのバージョンをチェックする。

【0041】

【表4】

```
$ cat prog.c
extern void foo1();
extern void foo2();

main ()
{
    foo1();
    foo2();
}

$ cc -o prog prog.c -L -R -lfoo
$ pvs -r prog
libfoo.so.1 (SUNW.1.2 SUNW.1.2.1);
```

【0042】図12は、再配置可能なオブジェクト118および共用オブジェクト114から動的に実行可能なELFファイル120を作成するために、構築時に前記リンクエディタ124によって実行されるステップを示すものである。前記動的に実行可能なELFファイル120のフォーマットは、図14に示されている。図14のフォーマットは、バージョン従属状態セクションを含むがバージョン定義セクションまたはバージョン記号セクションを含まないという点を除き、図9のフォーマットと同様である。図15は、バージョン従属状態セクションのフォーマットを示す図である。

【0043】図12のステップ802において、前記リンクエディタ124は、リンク中のオブジェクト（例えばprog）における未分解の大域記号を、リンク中の他のオブジェクト（例えばlibfoo1.so.1）の大域記号テーブルと照会することによって、前記オブジェクトprogが前記他のオブジェクトに従属しているか否かを判定する。従属している場合、ステップ804において、前記リンクエディタ124は、必要とされるオブジェクトが複数のバージョンを有するか否か、すなわち、前記必要とされるオブジェクトがバージョン定義セクションを有するか否かを判定する。ステップ806において、前記必要とされるオブジェクトのいくつかのバージョンのみが前記リンクエディタ124にとって可視性のものと判定された場合、制御はステップ810に進む。そうでない場合、制御はステップ808に進む。

【0044】ステップ808において、前記リンクエディタ124は、共用オブジェクト114のすべての利用可能なバージョン定義セクションを調べ、どのバージョンが再配置可能なオブジェクト118に必要な大域記号を含むかを判定することによって、動的に実行可能なオブジェクト120にバージョン従属状態セクションを作成する。または、マップファイル132が、いくつかのバージョンのみを、前記リンクエディタ124にとって可視性のものとして識別してもよい。ステップ810では、前記リンクエディタ124は、共用オブジェクト114の可視性のバージョン定義セクションを調べ、どのバージョンが再配置可能なオブジェクト118に必要な大域記号を含むかを判定する。

【0045】図14のバージョン定義セクションに記録される従属状態は、アプリケーションプログラムが共用

オブジェクトに対するインターフェイスの大域記号を参照する毎に作成される。表4は、アプリケーションプログラムソフトウェア116（“prog.c”）のソースバージョンの一例を示すものである。該アプリケーションプログラム“prog.c”は、SUNW.1.1バージョンに定義された大域記号“foo1”およびSUNW.1.2バージョンに定義された大域記号“foo2”（表2参照）を参照する。こうして、アプリケーションプログラムprog.cと（SUNW.1.1バージョンから受け継いだ）SUNW.1.2バージョンとの間には、従属関係が存在する。前記リンクエディタ124は、prog.cの動的に実行可能なオブジェクト120に、prog.cがSUNW.1.2バージョンに従属していることを示すバージョン従属状態セクションを作成する（図22参照）。前記再配置可能なオブジェクト118における未定義の大域記号毎に、前記リンクエディタ124は、前記再配置可能なオブジェクト118とリンクされているすべての共用オブジェクトのバージョン定義セクションおよびバージョン記号セクションを調べることによって、前記記号がどのバージョンに属するものかについての情報を得る。

【0046】図15は、バージョン定義セクション510のフォーマットを示すものである。該バージョン定義セクション510は、図16に関連して説明するセクションヘッダ1102と、構造バージョン1104と、カウント値1106と、ファイル名1108と、補助値1110と、次バージョン従属状態セクション1112と、複数のフィールド1114～1122のインスタンスとを含んでいる。前記フィールド1114～1122は、ハッシュ値1114と、“ウィーク”フラグ1116と、未使用フィールド1118と、名前フィールド1120と、次名前フィールド1122とを含んでいる。

【0047】図10に関連して上述したように、構造バージョン値1104はこの発明のバージョン設定に関係しない。カウント値1106は、フィールド1114～1122のインスタンスの数を示す。ファイル名1108は、共用オブジェクトの従属状態の名前である。補助値1110は、このバージョンに関する第1のフィールド1114に対する索引である。次バージョン従属状態セクション1112は、次のバージョンフィールド1104に対する索引（またはゼロ）である。前記フィールド1114～1122の各インスタンスは、作成中のオブジェクトが必要とするバージョンを示す。バージョンが前記オブジェクトによって参照される大域記号を定義する場合、該バージョンが必要になる。常に、前記フィールド1114～1122の少なくとも1つのインスタンスが存在する。

【0048】ハッシュ値1114は、バージョン名1120から、通常のELFハッシュ関数を使用して発生される。“ウィーク”フラグ1116は、被従属（必要とされる）バージョンが弱いバージョンか否か、すなわち、

該バージョンが大域記号を含んでいないものか否かを示す。フィールド1118は、この発明のすべての実施の形態において形成されるものではなく、単に位置合せのために含まれる。ネームフィールド1120の第1のインスタンスは、前記バージョン自体の名前(例えば、“SUNW.1.2”)を含むものである。次名前フィールド1122は、次のハッシュ値1114を示すポインタである。

【0049】この発明のこの実施の形態は、すべての継承(受け継いだ情報)がバージョン定義セクション510に記録されないようにする、“バージョン縮小化”を使用する。弱バージョンおよび必要とされるバージョンが、記録される。しかし、これらのバージョンが他のバージョンから大域記号を継承する名前、一般に、前記他のバージョンは記録されない。例えば、第1の弱いバージョンからの第2のバージョンを介した継承は、フィールド1120、1122に記録される前に、縮小化されるのが好ましい。同様に、第1の弱いバージョンからの第2の弱いバージョンを介した継承も、記録される前に縮小化されるのが好ましい。弱いバージョンと弱いバージョンとの間の継承は、縮小化されない。こうして、SUNW.1.2はフィールド1120、1122に記録されるが、SUNW.1.1からの継承は縮小化されない。さらに、弱いバージョンであるSUNW.1.2.1に対する従属状態が記録される。

【0050】図16〜図18は、図10、図11および図15のセクションの様々な様相を示すものである。図16〜図18のフィールドは、従来のELFフォーマットの一部であるので、ここではこれらのすべては説明せず、この発明に関係するフィールドのみについて説明する。図16は、セクションヘッダ607、702、1102に使用されるセクションヘッダのフォーマットを示す図である。“sh\_name”フィールド1202はセクションの名前を含み、“sh\_type”フィールド1204は前記セクションの種類を含んでいる。前記フィールドの値1202は、特にこの発明に関連のあるものである。図17は、前記フィールド1204の様々な種類を表す

値を示すものである。値1302は、特にこの発明に関連のあるものである。図16のセクションヘッダフォーマットは、さらに、前記セクションにおけるバット数を示す“sh\_size”フィールド1306と、“sh\_link”フィールド1308と、“sh\_info”フィールド1310とを含んでいる。図18は、前記“sh\_type”、“sh\_link”および“sh\_info”の値の例を示している。値1402は、特にこの発明に関連のあるものである。

【0051】上記パラグラフは、構築時における、バージョン定義セクション506およびバージョン記号セクション508を含むELFオブジェクト1140の作成、および、バージョン従属状態セクション510を含むELFオブジェクト120の作成について説明した。なお、前記バージョン定義セクション506およびバージョン記号セクション508は大域記号の定義を含むオブジェクトのみに作成されるものであり、バージョン従属状態セクション510は、大域記号を含むオブジェクトに属するオブジェクトのみに作成されるものである。こうして、大域インターフェースを有する共用オブジェクトは、セクション506、508および510を含むことになる。前記共用オブジェクトを参照する実行可能なアプリケーションプログラムは、バージョン従属状態セクション510のみを含むことになる。

【0052】c. 構築時におけるアプリケーション構造の制限  
上記パラグラフは、共用オブジェクト114のすべての利用可能なバージョンと再配置可能なオブジェクトとの間のバージョン結合の一例について説明した。結合は、アプリケーションプログラムと、図13のオブジェクトの特定のバージョンのみとの間で発生するよう制限されることもできる。図13は、図4のマッピングファイル132におけるファイル制御指示のフォーマットを示すものである。このフォーマットのファイル制御指示は、他のオブジェクトの特定のバージョンに従って動的に実行可能なオブジェクトを結合するために使用される。

【0053】

【表5】

```
$ cat mapfile
libfoo.so - SUNW.1.1;

$ cc -o prog prog.o -M mapfile -L . -R -lfoo
Undefined               first referenced
  symbol
foo2                      prog.o (symbol belongs to \
unavailable version ./libfoo.so (SUNW.1.2))
ld: fatal: Symbol referencing errors. No output written to prog
```

【0054】図4および表5に示すように、前記リンクエディタが図13に示されたファイル制御を含むマッピングファイル132を使用して実行される場合、前記オブジェクトの特定のバージョンについてのみ結合が行われる。表5において、マッピングファイルは、“libfoo.so-SUNW.1.1.”を含んでいる。表4に示すように、アプリ

ケーションソフトウェア“prog.o”は、大域記号“foo1”および“foo2”を参照する。記号“foo2”は、SUNW.1.2バージョンに定義されている。(ccコンパイラおよびld(1)リンクエディタを使用して) prog.oが-Mオブジェクトとリンクされる場合、該prog.oは、SUNW.1.1バージョンのみとリンクされる。このようにして、表5に示

すように、リンクエディタ 124 は、SUNW.1.2バージョンに定義された“foo2”を未定義の記号として認識する。

【0055】d. 構築時における弱いバージョンの昇格表6は、リンクエディタ 124 が弱いバージョン (SUNW.1.2.1) を強いバージョンに“昇格”するよう強制される例を示すものである。“-u”オプションにより、リンクエディタ 124 は、SUNW.1.2.1バージョンに対する“prog”の従属状態を、該“prog”のバージョン従属状態セクションに記録する。さらに、“ウィーク”フラグ 1116 が、SUNW.1.2.1バージョンに関して、“prog”のバージョン従属状態セクションにおいて、“弱”に設定される。この場合、SUNW.1.2.1バージョンはSUNW.1.2バージョンを受け継ぐ。故に、progの従属状態のすべては“強”であるので、バージョンの縮小化は、SUNW.1.2.1バージョンのみが弱いバージョンとしてprogに記録される、ことを意味する。このような弱いバージョンの昇格は、“prog”が実行される時、それまで弱いものであったバージョンが存在することが実行時リンクエディタ 124 によって確認される、ことを保証する。

【0056】

【表6】

```
$ cc -o prog prog.c -L. -R. -u SUNW.1.2.1 -lfoo
```

```
$ ldd -r prog  
libfoo.so.1 (SUNW.1.2.1)
```

【0057】3. 実行時におけるバージョン設定情報の確認

図19は、動的に実行可能なオブジェクト 120 をリンクして実行するときに、参照されるオブジェクトの必要なバージョンのすべてが存在することを保証するために、実行時リンカ 126 (図5参照) によって実行されるステップを示すフローチャート 1500 である。図19のステップは、好ましくは、CPU 102 がメモリに格納された命令を実行することによって行われる。図19のステップは、必要な従属状態のすべてが存在しているかを判定するために、実行前チェックとして行われる。

【0058】ステップ 1502 において、前記リンカ 126 は、実行中の動的に実行可能なオブジェクト 120 がバージョン従属状態セクション (図15参照) を含んでいるかを判定する。この判定結果がNOである場合、なんらかのチェックはなされず、通常の実行が継続される。この判定結果がYESである場合、ステップ 1504 において、前記リンカ 126 は、リンク中のオブジェクト 114 の少なくとも一つがバージョン定義セクションおよびバージョン記号セクションを含んでいるかを判定する。この判定結果がNOである場合、それ以上のチェックはなされず、通常の実行が継続される。この

判定結果がYESである場合、制御はステップ 1506 に進む。

【0059】ステップ 1506 において、前記実行時リンカ 126 は、現在の動的に実行可能なオブジェクト 120 におけるすべてのバージョン従属状態が処理されたかを判定する。すべてのバージョン従属状態が処理された場合、通常の実行が継続され、そうでない場合、制御はステップ 1508 に進む。ステップ 1508 では、前記実行時リンカ 126 は、(前記動的に実行可能なオブジェクト 120 のバージョン従属状態セクション) 必要とされるバージョンと、前記共用オブジェクト 114 のバージョン定義セクションとが一致するか否かを判定する。例えば、図20および図22において、SUNW.1.2バージョンおよびSUNW.1.2バージョンは、共用オブジェクト 114 のバージョン定義セクションに定義されており、必要に応じて、動的に実行可能なオブジェクト 120 のバージョン従属状態セクションにおいて指定される。

【0060】ステップ 1508 において一致が検出された場合、共用オブジェクト 114 の必要とされるバージョンが存在していることになり、ステップ 1508 において一致が検出されなかった場合、必要とされるバージョンが存在していないことになるので、ステップ 1510 において、その存在していないバージョンが弱いバージョンであるか、または、弱いバージョンであるかを判定することが必要になる。前記実行時リンカ 126 は、動的に実行可能なオブジェクト 120 のバージョン従属状態セクションの“ウィーク”フラグ 1116 をチェックすることによって、前記バージョンが弱いものかを判定する。(なお、前記フラグ 1116 は、バージョンの弱から強への昇格を反映してもよい)。前記実行時リンカ 126 が、ステップ 1510 において、そのバージョンが弱いバージョンであると判定した場合、エラーは発生せず、制御はステップ 1506 に戻る。弱いバージョンではない判定した場合、必要とされる弱いバージョンが存在していないので、致命的なエラーが発生する。従って、存在していない“弱い”バージョンへの従属はエラーを発生しないが、“弱くない”バージョンへの従属はエラーを発生することになる。

【0061】図20は、バージョン定義セクション 1600 の一例を示し、図21は、共用オブジェクト 114 (表1～表3のlibfoo.so.1) に関するバージョン記号セクション 1700 の一例を示す。さらに、図22は、アプリケーションプログラム 120 (表4の“prog”) の動的に実行可能な形態に関するバージョン従属状態セクション 1800 の一例を示すものである。(この例において、共用オブジェクト 114 は、他のオブジェクトに従属しており、従って、バージョン従属状態セクションを有さない)。

【0062】図20は、基準バージョン定義 1604、

および、それぞれのバージョンSUNW. 1. 1, SUNW. 1. 2, SUNW. 1. 2. 1, SUNW. 1. 3a, SUNW. 1. 3bおよびSUNW. 1. 4のための6つのバージョン定義を含んでいる。前記基準バージョン定義1604の“基準”フラグは、“真(true)”に設定されている。また、SUNW. 1. 2. 1バージョンに関するバージョン定義の“弱”フラグは、“真(true)”に設定されている。各バージョン定義は、固有のバージョン定義索引を有する。バージョンの縮小化は、バージョン定義テーブルにおいて適用される。アプリケーション“prog”は、(SUNW. 1. 1)の大域記号fooおよび(SUNW. 1. 2)の大域記号foo2を参照する。SUNW. 1. 2はSUNW. 1. 1の大域記号を受け継ぐので、前記リンクエディタ124はバージョン縮小化を適用し、(SUNW. 1. 2のための)エントリがバージョン定義セクションにおいてなされる。弱いSUNW. 1. 2. 1バージョンも記録される。

【0063】図21は、バージョン記号セクション170における記号のうちのいくつかを示す図である。例えば、大域変数“foo1”は、SUNW. 1. 1バージョンに定義されており、バージョン定義索引“2”を有する。こうして、foo1に関する記号テーブルのエントリに対応する前記バージョン記号セクションのエントリは、“2”を含む。上記実施の形態において、各バージョン(例えば、SUNW. 1. 1)の名前は、そのバージョン定義を生成するときに作成される大域記号でもある。

【0064】図22は、動的に実行可能なオブジェクト120におけるバージョン従属状態セクションの一例を示す図である。バージョン縮小化により、前記セクションは、SUNW. 1. 1ではなく、SUNW. 1. 2のためのエントリを含む。該セクションは、“弱い”SUNW. 1. 2. 1バージョンのためのエントリをも含む。実行時において、リンク126は、progがバージョン従属状態セクションを有すると判定し(ステップ1502)、libfoo.so.1がバージョン定義セクションを有すると判定し(ステップ1504)、必要とされるバージョン(SUNW. 1. 2)が存在すると判定する(ステップ1508)。故に、“prog”が実行されることになる。以上、いくつかの好ましい実施例について本発明につき説明したが、これら実施例に限らず、本発明の精神と範囲を逸脱しない限り、その他の種々の変形が可能であることが理解されるであろう。

【0065】最後に、本願に係わる発明及びその実施態様のいくつかを要約して示すと下記の通りである。

(1) ソフトウェアプログラムにバージョン設定情報を付す方法であって、第1のソフトウェアプログラムのための第1のオブジェクトコードを用意するステップと、前記第1のソフトウェアプログラムのバージョンに関するバージョン名を示すマップファイルを用意するステップと、前記第1のソフトウェアプログラムのバージョン名を示す情報が、前記マップファイルに従って前記第1のオブジェクトコードに付加されるよう、前記第1のオブジェクトコードをリンクし、これにより、バージョン

設定されたオブジェクトを生成するステップとを具備し、これらのステップがデータ処理システムによって実行される方法。

(2) 第2のソフトウェアプログラムのための第2のオブジェクトコードを用意するステップと、前記第2のオブジェクトコードに前記バージョン設定されたオブジェクトにリンクするステップとをさらに具備し、前記リンクするステップが、前記第2のソフトウェアプログラムに必要とされる前記第1のソフトウェアプログラムのバージョンを判定するステップと、前記第2のソフトウェアプログラムに必要とされる前記バージョンを示す情報を前記第2のオブジェクトコードに付加し、これにより、動的に実行可能なプログラムを生成するステップとをさらに含むことを特徴とする前記1項に記載の方法。

(3) 前記動的に実行可能なプログラムを実行する前に、該実行可能なプログラムにおいて必要とされるバージョンが、前記バージョン設定されたオブジェクトにおいて示されるバージョンと一致するか否か、を判定するステップと、前記動的に実行可能なプログラムを実行するステップとをさらに具備し、前記動的に実行可能なプログラムが前記バージョン設定されたオブジェクトの前記必要とされるバージョンを呼出す前記2項に記載の方法。

(4) 前記マップファイルが、前記第1のソフトウェアプログラムの前記バージョンのインターフェースを構成する大域記号をさらに指定し、前記生成するステップが、前記バージョン設定されたオブジェクトを生成するために、前記マップファイルに従って、前記バージョン名、および、前記バージョンのインターフェースを構成する大域記号を示す情報を前記第1のオブジェクトコードに付加するステップを含む前記1項に記載の方法。

(5) 前記判定するステップが、どの大域記号が前記第2のソフトウェアプログラムに必要とされるかを調べることによって、前記第2のソフトウェアプログラムに必要とされる前記第1のソフトウェアプログラムのバージョンを判定し、さらに、前記バージョン設定されたオブジェクトにおける情報をチェックすることによって、前記必要とされる大域記号がどのバージョンに存在するのかを判定するステップを含む前記2項に記載の方法。

(6) 前記第1のオブジェクトコードに付加される情報がバージョン定義セクションである前記1項に記載の方法。

(7) 前記第1のオブジェクトコードに付加される情報がバージョン記号セクションである前記1項に記載の方法。

(8) 前記第1のオブジェクトコードに付加される情報がバージョン従属状態セクションである前記2項に記載の方法。

(9) 前記バージョン設定されたオブジェクトが、再

配置可能なオブジェクトである前記1項に記載の方法。

(10) 前記バージョン設定されたオブジェクトが、動的に実行可能なオブジェクトである前記1項に記載の方法。

(11) 前記バージョン設定されたオブジェクトが、共用オブジェクトである前記1項に記載の方法。

【0066】(12) ソフトウェアプログラムにバージョン設定情報を付す装置であって、第1のソフトウェアプログラムのための第1のオブジェクトコードを格納する記憶媒体と、前記第1のソフトウェアプログラムのバージョンに関するバージョン名を指定するマップファイルを格納する記憶媒体と、前記マップファイルに従って、前記第1のオブジェクトコードに対して、前記第1のソフトウェアプログラムの前記バージョンのバージョン名を定義する付加情報を付し、これにより、バージョン設定されたオブジェクトを生成するリンクとを具備する装置。

(13) 第2のソフトウェアプログラムのための第2のオブジェクトコードを格納する記憶媒体と、前記第2のソフトウェアプログラムに必要とされるバージョンを示す付加情報を前記第2のオブジェクトコードに付すことによって、前記第2のオブジェクトコードを前記バージョン設定されたオブジェクトにリンクし、これにより、動的に実行可能なプログラムを生成するリンクとをさらに具備した前記12項に記載の装置。

(14) 前記動的に実行可能なオブジェクトにおいて指定された前記必要とされるバージョンが、前記バージョン設定されたオブジェクトに定義されたバージョンに一致していると判定した場合、前記動的に実行可能なプログラムの実行を可能にする実行時リンクをさらに具備した前記13項に記載の装置。

(15) 前記バージョン設定されたオブジェクトが、再配置可能なオブジェクトである前記12項に記載の装置。

(16) 前記バージョン設定されたオブジェクトが、動的に実行可能なオブジェクトである前記12項に記載の装置。

(17) 前記バージョン設定されたオブジェクトが、共用オブジェクトである前記12項に記載の装置。

【0067】(18) 動的に実行可能なオブジェクトに必要とされるオブジェクトのバージョンが該実行可能オブジェクトの実行中に存在することを判定させるための、コンピュータによって読み取り可能なコードを格納したコンピュータによって使用可能な媒体を備えたコンピュータプログラム製品であって、コンピュータに、第1のソフトウェアプログラムのための第1のオブジェクトコードを用意させる第1のコンピュータ読み取り可能プログラムコード装置と、コンピュータに、前記第1のソフトウェアプログラムのバージョンに関連したバージョン名を指定するマップファイルを用意させる第2のコ

ンピュータ読み取り可能プログラムコード装置と、前記第1のソフトウェアプログラムのバージョン名を示す情報が、前記マップファイルに従って、前記第1のオブジェクトコードに付加されるよう、コンピュータに、前記第1のオブジェクトコードをリンクさせ、これにより、バージョン設定されたオブジェクトを生成させる第3のコンピュータ読み取り可能プログラムコード装置とを具備したコンピュータプログラム製品。

(19) コンピュータに、第2のソフトウェアプログラムのための第2のオブジェクトコードを用意させる第4のコンピュータ読み取り可能プログラムコード装置と、コンピュータに、前記第2のソフトウェアプログラムに必要とされる前記第1のソフトウェアプログラムのバージョンを判定させ、前記第2のソフトウェアプログラムに必要とされるバージョンを示す情報を前記第2のオブジェクトコードに付加させることによって、前記バージョン設定されたオブジェクトに対する前記第2のオブジェクトコードのリンクを実行させる第5のコンピュータ読み取り可能プログラムコード装置とをさらに具備した前記18項に記載のコンピュータプログラム製品。

(20) コンピュータに、前記動的に実行可能なプログラムがプログラムを実行するために必要とされるバージョンが前記バージョンを有するオブジェクトに定義されたバージョンと一致しているか否かを判定させる第6のコンピュータ読み取り可能プログラムコード装置をさらに具備し、前記動的に実行可能なプログラムが前記バージョン設定されたオブジェクトの前記必要とされるバージョンを呼出すことを特徴とする前記19項に記載のコンピュータプログラム製品。

【0068】

【発明の効果】以上のように、この発明は、オブジェクトの新たなバージョンが作成される毎に該オブジェクトの名前を変える必要性を無くし、且つ、バージョン変更に対処するために必要な部分のみのアップグレードを可能にするので、簡単且つ効率的なバージョン設定を実現できる、という優れた効果を実現する。

【図面の簡単な説明】

【図1】この発明に係るコンピュータシステムを示すブロック図。

【図2】構築時における図1のリンクエディタの入出力を示す図。

【図3】構築時における図1のリンクエディタの他の入出力を示す図。

【図4】構築時における図1のリンクエディタの他の入出力を示す図。

【図5】実行時における図1の実行時リンクプログラムの入出力を示す図。

【図6】オブジェクトにバージョン定義セクションおよびバージョン記号セクションを付加するために、図2のリンクエディタによって実行されるステップを示すフロ



ーチャート。

【図7】図6の処理の詳細を示すフローチャート。

【図8】図2のマップファイルのフォーマットを示す図。

【図9】バージョン設定されたオブジェクトに含まれる図2のリンクエディタの出力を示す図。

【図10】図9のバージョン定義セクションのフォーマットを示す図。

【図11】図9のバージョン記号セクションのフォーマットを示す図。

【図12】動的に実行可能なアプリケーションプログラムに含まれる図3のリンクエディタの出力を示すために、図3または図4のリンクエディタによって実行されるステップを示すフローチャート。

【図13】図4のマップファイルのフォーマットを示す図。

【図14】動的に実行可能なアプリケーションプログラムに含まれる図3のリンクエディタの出力を示す図。

【図15】図9および図14のバージョン従属状態セクションのフォーマットを示す図。

【図16】図10、図11および図15のセクションヘッダのフォーマットを示す図。

【図17】図16のヘッダにおける様々な値のリストを示す図。

【図18】図16のヘッダにおける様々な値のリストを示す図。

【図19】アプリケーションプログラムのバージョン要件が該アプリケーションプログラムにリンクされているオブジェクトに存在するバージョンに一致していることを確認するために、図5の実行時リンカによって実行されるステップを示すフローチャート。

【図20】前記リンクエディタによってオブジェクトに付加されたバージョン定義セクションの一例を示す図。

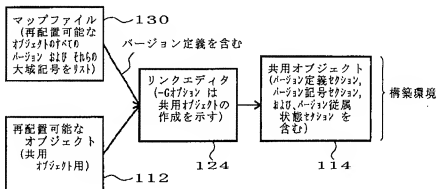
【図21】前記リンクエディタによってオブジェクトに付加されたバージョン記号セクションの一例を示す図。

【図22】前記リンクエディタによってバージョン設定されたオブジェクトおよび動的に実行可能なアプリケーションプログラムの一方または両方に付加可能なバージョン従属状態セクションの一例を示す図。

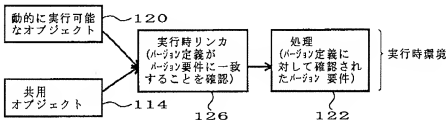
【符号の説明】

- 102 CPU
- 104 メモリ
- 110 ソースコード
- 114 共用オブジェクト
- 116 ソースコード
- 118 再配置可能なオブジェクト
- 120 動的に実行可能なオブジェクト
- 124 リンクエディタ
- 124 実行時リンカ
- 130 マップファイル
- 132 マップファイル

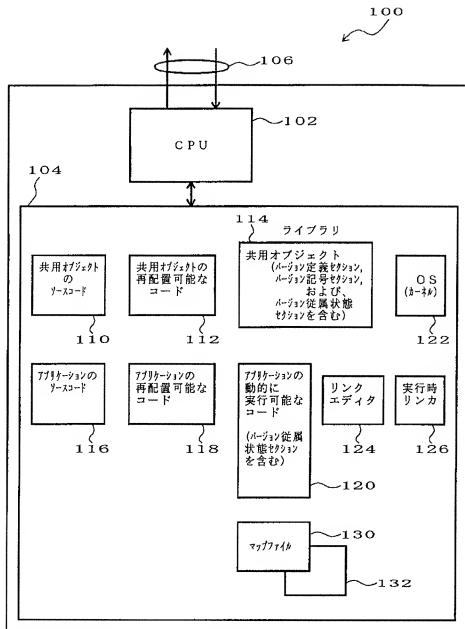
【図2】



【図5】



【図1】



【図8】

```
[<version name>] {
  [<scope>]
  [<symbol(s)>];
} [<inheritance information>];
```

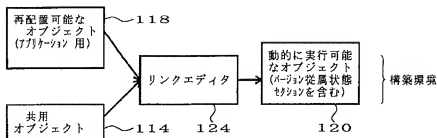
(v2.77(64bit)) リンク指示のフォーマット (バージョン定義)

【図13】

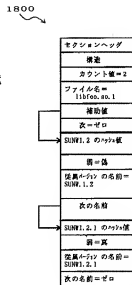
```
<object: name>-<version(s)>:
```

リンク指示のフォーマット  
(マップファイルにおけるバージョン解凍指示)

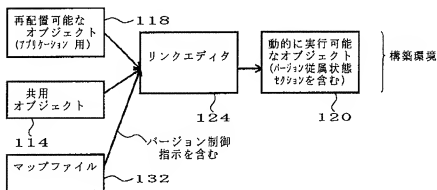
【図3】



【図22】

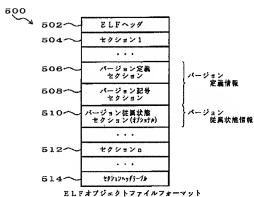


【図4】



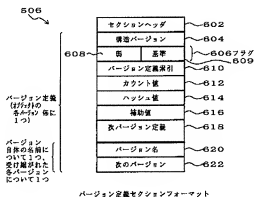
バージョン従属セクション

【図9】



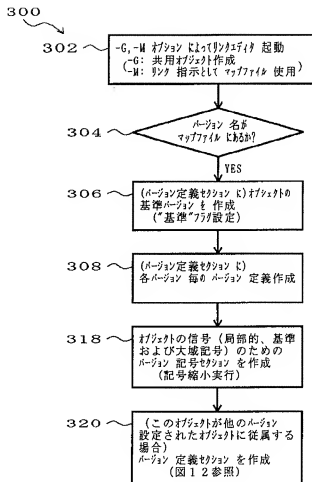
ELFオブジェクトファイルフォーマット

【図10】



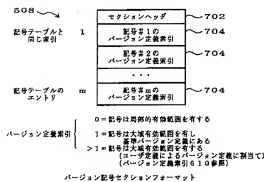
バージョン定義セクションフォーマット

【図6】



リンクエディタがバージョン定義セクション  
およびバージョン記号セクションを作成する処理

【図11】



【図16】

セクションヘッダフォーマット

```
typedef struct {
    Elf32_Word  sh_name;  ~1202
    Elf32_Word  sh_type;  ~1204
    Elf32_Word  sh_flags;
    Elf32_Addr  sh_addr;
    Elf32_Off   sh_offset;
    Elf32_Word  sh_size;  ~1306
    Elf32_Word  sh_link;  ~1308
    Elf32_Word  sh_info;  ~1310
    Elf32_Addr  sh_addrlocal;
    Elf32_Word  sh_entsize;
} Elf32_Shdr;
```

【図17】

セクションタイプ, sh\_type

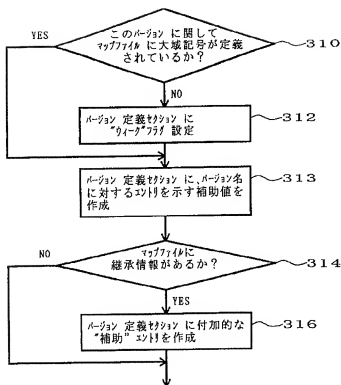
名前	値
SHT_NULL	0
SHT_PROGBITS	1
SHT_SYMTAB	2
SHT_STRTAB	3
SHT_RELA	4
SHT_HASH	5
SHT_DYNAMIC	6
SHT_NOTE	7
SHT_NOBITS	8
SHT_REL	9
SHT_SHLIB	10
SHT_DYNSYM	11
SHT_SUNW_verdef	0x00000000
SHT_SUNW_verneed	0x00000001
SHT_SUNW_versym	0x00000002
SHT_LOPROC	0x70000000
SHT_HIPROC	0xffffffff
SHT_LOUSER	0x80000000
SHT_HIUSER	0xffffffff

【図14】



ELFオブジェクトファイル  
(リンクエディタからの動的に実行可能な出力)

【図7】

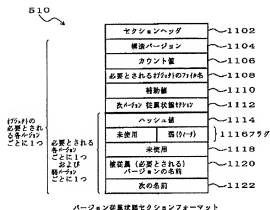


バージョン定義セクション作成の詳細

【図21】

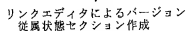
バージョン定義セクション (libfoo.so.1)	記号タイプ (libfoo.so.1)
セクションヘッダ	data
1	...
2	SYMBOL.1.1
...	...
2	foo1
...	...
3	foo2
...	...
0	Global symbol1
...	...
5	bar1
...	...
6	bar2
...	...
7	bar3
...	...

【図15】

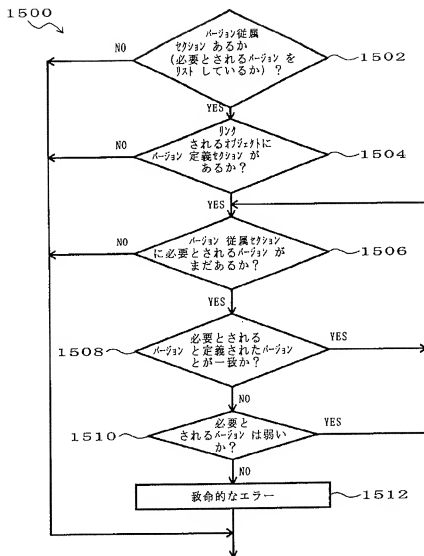


【図18】

sh_link および sh_info の説明		
sh_type	sh_link	sh_info
SHT_DYNAMIC	継承バージョン # セクションヘッダ参照	0
SHT_HASH	継承バージョン # セクションヘッダ参照	0
SHT_REL SHT_RELA	継承バージョン # セクションヘッダ参照	消費型が適用される セクションのセクションヘッダ参照
SHT_STRTAB SHT_DYNAMIC	継承バージョン # セクションヘッダ参照	最後の局所的記号の記号 テーブル参照より大きいもの (SHT_LOCAL 結合)
SHT_SYMTAB SHT_DYNAMIC	継承バージョン # セクションヘッダ参照	セクション内のセクション の定義の値
SHT_SYMTAB SHT_DYNAMIC	継承バージョン # セクションヘッダ参照	セクション内のセクション の定義の値
SHT_SYMTAB SHT_DYNAMIC	継承バージョン # セクションヘッダ参照	0
SHT_SYMTAB SHT_DYNAMIC	継承バージョン # セクションヘッダ参照	0
その他	SEN_UNDEF	0

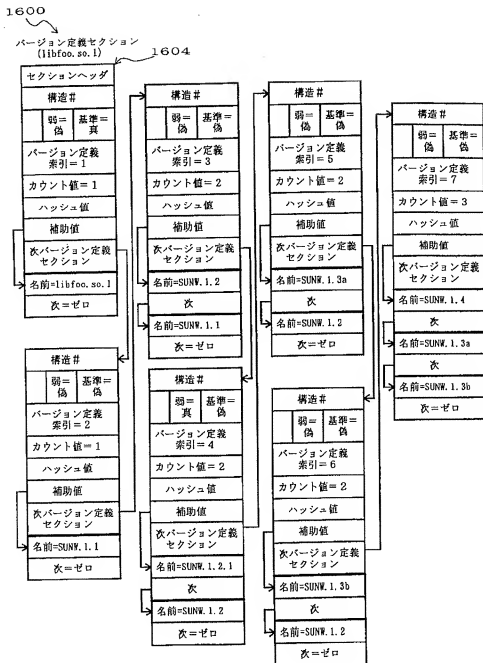


【図19】



実行時確認

【図20】



フロントページの続き

(72)発明者 ロバート エー ジンゲル  
 アメリカ合衆国 94087 カリフォルニア,  
 サニーベイル, ライトアベニュー, 1377